World Journal of Advanced Engineering Technology and Sciences
eISSN: 2582-8266
Cross Ref DOI: 10.30574/wjaets
Journal homepage: https://wjaets.com/

(RESEARCH ARTICLE)

Check for updates

Autonomous cars in rural area roads

ST Patil, Ashwini Patil, Sayali Jitendra Pophale *, Ritesh Jogendra Prajapati, Prateek Sunil Puranik and Rushabh Pawan Potarwar

Vishwakarma Institute of Technology, Pune, Maharashtra, India.

World Journal of Advanced Engineering Technology and Sciences, 2022, 07(02), 180–188

Publication history: Received on 26 October 2022; revised on 04 December 2022; accepted on 06 December 2022

Article DOI: https://doi.org/10.30574/wjaets.2022.7.2.0139

Abstract

Vehicles must be able to perceive their surroundings like human drivers in order to navigate roadways, wait at traffic lights and stop signals to avoid colliding with barriers like other vehicles and pedestrians. Focusing on the difficulties encountered by autonomous vehicles in recognising objects, a method has been developed to lane detection and tracking with the help of OpenCV library. The purposes and techniques towards using grayscale rather than colour, identifying edges in an image, choosing the zone of interest, implementing the Hough Transform, and using polar coordinates rather than Cartesian coordinates have all been addressed. The Canny Edge detection algorithm for lane detection is being implemented on a self-driving automobile prototype built on the Raspberry Pi. Our approach seeks to enable smooth lane detection for a quick and precise response to lane changes on the road. The implementation's difficulties include assessing the road and lane borders, as well as predicting the proper degree of rotation of the wheel motor while keeping the vehicle's centre in line with the frame centre.

Keywords: Numpy; OpenCV; Canny; Lane Detection; Hough Transform; Mat Plot Lib; Gaussian Blur

1. Introduction

While driving, humans use their optical vision to vehicle navigate their vehicle. The lane markings on the road serve as a continual point of reference for moving vehicles. One of the necessities for a self-driving car is the evolution of an algorithm-based Automatic Lane Detection .Computer vision allows automobiles to perceive their surroundings. It is a subset of artificial intelligence which enables software to comprehend the information of photos and videos. Deep learning breakthroughs have enabled modern computer vision to detect distinct elements in photos by evaluating and comparing thousands of instances and identifying the visual patterns that distinguish each object. Although deep learning is very effective for classification tasks, it has major limits and can fail in unanticipated ways. For example, in broad daylight, an autonomous vehicle may collide with a truck, or worse, strike a person. The existing computer vision technology employed in driverless cars is also susceptible to adversarial attacks, which compel the AI to commit mistakes by manipulating its input channels. For example, the previously stated challenges caused from changes in lane boundaries. The approach employed in this research is intended to detect lane markings by inputting a video of the road into the system through computer vision technology, with the foremost objective of lowering the frequency of accidents. Road accidents can be prevented by implementing a system in automobiles. It is implemented in school buses to assure the safety of the pupils. Additionally, the driver's performance can be monitored, and RTO can utilise the technology to track and report driver's lack of attention and recklessness on the roadways.

2. Literature review

Numerous image processing research studies have focused on lane detection. A review of these studies reveals that they can possibly be categorized into two distinct categories. The first employs the bird's-eye view transform in a rear-view

* Corresponding author: Sayali Pophale

camera input image to detect lane markings [3]. The second category employs a front-facing camera. In this latter example, various image processing algorithms have been developed, including the Likelihood of Image Shape (LOIS) algorithm [4, B-Snake algorithm [5, and others [6, 7] that uses a feature-based approach to extract features present in an image (e.g. edges) with different methodologies.

A feature-based method has the benefit of providing a wide number of strategies that can be employed in each phase of the algorithm and therefore contribute to its optimization. However, it has two significant flaws. The first challenge of lane detection is the amount of computation needed. Therefore, for real world applications, the above-mentioned algorithm's performance and processing time can be enhanced by employing additional operators for edge detection and line tracking. The second challenge is that detecting lane boundary lines is complex since a real-world road picture may contain other possible edges (e.g., noise edges). As a result, selecting an edge detection approach wisely as to limit the amount of edge points in the output image of this stage. Indeed, for the Hough transform to be efficient, the edges must be identified well.

3. Proposed methodology

The problem statement for this paper questions a self-driven vehicle's potential to detect, track, and shift lanes on a roadway, identify impediments, as well as keeping a safe distance from other cars. The project's workflow begins with installing the OpenCV libraries on the Raspberry Pi and then gathering photos using the HD megapixel installed on the prototype's top. The frame size is used to specify an area of interest, and the points in the BGR scheme for the histogram pixels are defined as x-y coordinates. The image is processed and transformed from RGB to Grayscale before being fed to an algorithm that visualises the frequency of data arriving over a particular interval. This information is passed to canny edge detection algorithm, which employs the contrast gradients to spot out road and lanes. The edge detection technique constructs frame of lanes and travel route. The lane centre, which employs the left and right lane locations, is defined as a variable. The relative difference between the lane and centre of the frame is used to determine the amount of rotation/turn necessary to remain in the middle of the road. The Arduino receives this difference variable through Parallel Communication and operate the motor drivers to achieve proper speed and steering. This procedure is continued till the automobile arrives at its destination.

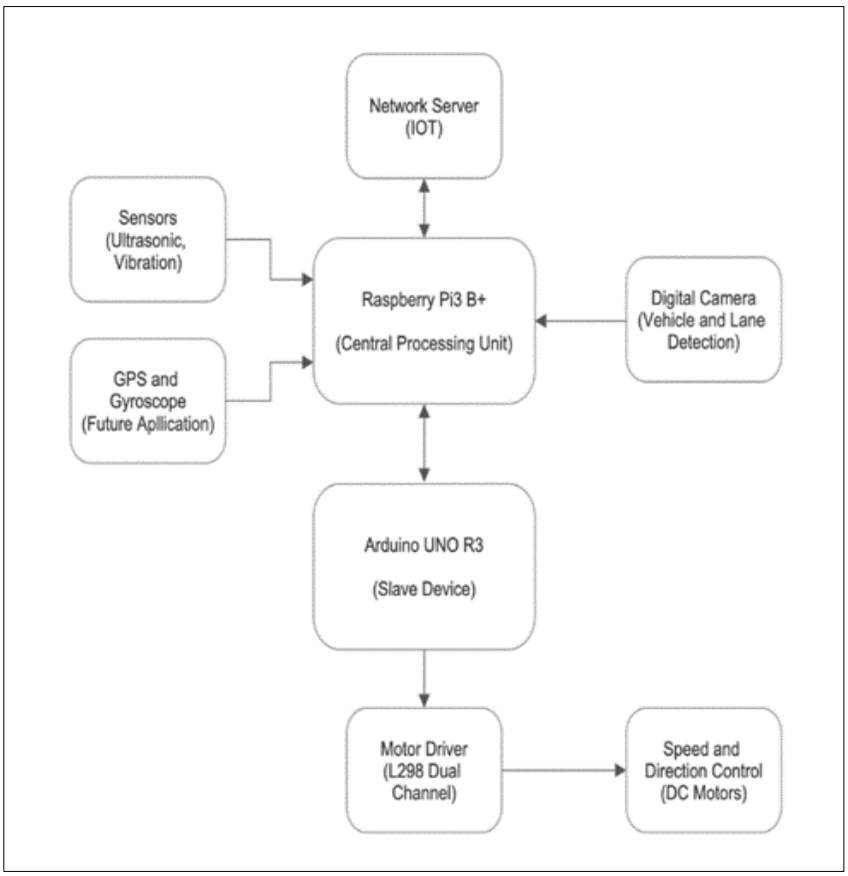

Figure 1 Block Diagram

The project entails detecting lanes in an image with the help of Python and OpenCV. OpenCV stands for "Open Source Computer Vision," and it is a package of image analysis tools.

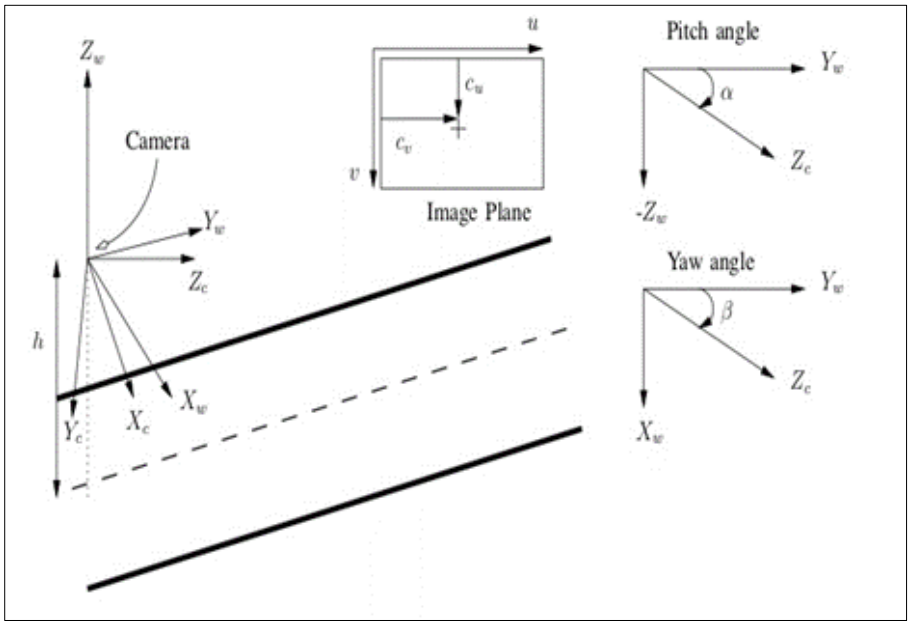

Figure 2 Angle detection

3.1. The Canny Edge Detection Technique

Edge detection Approach tries to determine edges of objects inside images. A detection algorithm is applied in an attempt to locate regions in a picture with a sudden shift in intensity. An image is represented as a matrix or array of pixels. A pixel corresponds to the light intensity at a distinct point in an image Every pixel's intensity is represented by a number ranging from 0 to 255; zero implies no light intensity when something is fully black, whereas 255 corresponds to maximum light intensity when something is completely white. A gradient is a change in brightness over a range of pixels. A steep shift is represented by a strong gradient, whereas a shallow change is represented by a short gradient.

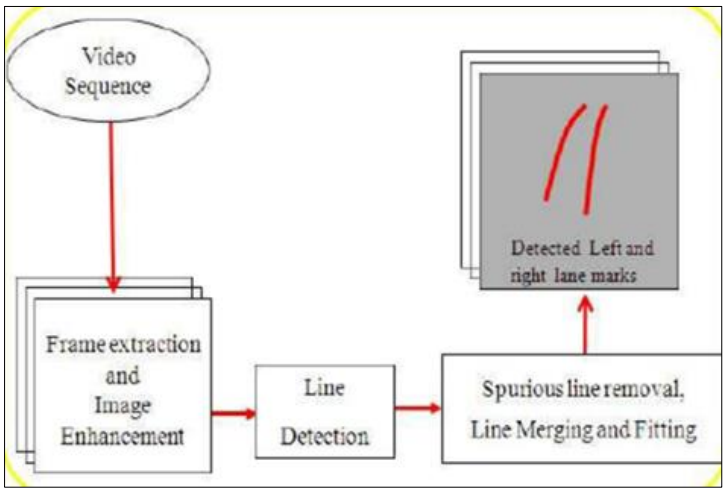

Figure 3 Curve and Line detection System

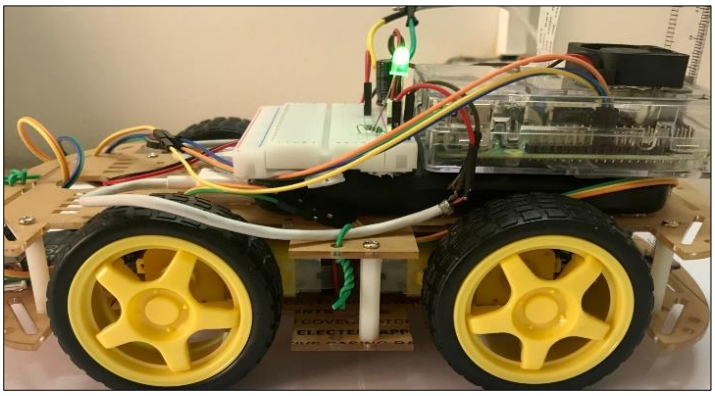

Figure 4 Autonomous Car

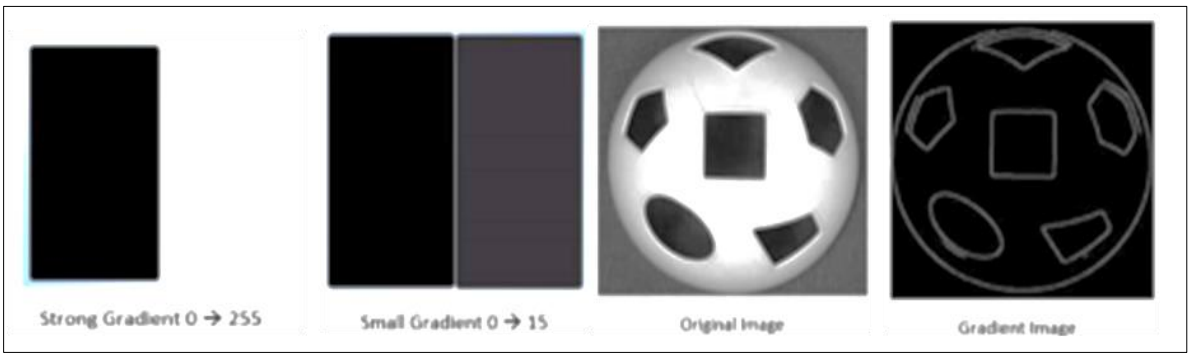

Figure 5 Canny Edge Detection System

Above figure represents the gradient of the soccer ball. The brightness discontinuity at the locations where the gradient is highest corresponds to the contour of white pixels. This aids in identifying edges in our image because an edge is defined by the difference in intensity levels between neighbouring pixels. Additionally, every time there is a steep gradient, or rapid shift in brightness, there is an identical bright pixel in the gradient image. Tracing out each of these pixels yields the edges. We'll apply this concept to find the edges in our road image.

Figure 6 Road and Lanes

To transform the image into grayscale, initially we will make use of Numpy to produce a clone of the original image:

An image with three colour channels would have an RGB channel, every pixel in the image is a blend of three intensity values. Each pixel in a grayscale image has only one intensity value that ranges from 0 to 255.

RGB image converted to grayscale.

Figure 7 Gaussian blur

Each pixel in a grayscale picture is characterised by a single integer that represents its brightness. To smoothen a picture, the traditional solution would be to replace each pixel's value with average value of the intensities of the pixels around it. In order to reduce noise, a kernel averages the pixels. Kernel numbers (np.array([[1,2,3],[

,5,6],[7,8,9]]) are applied to the entire image, setting each pixel to the pixels near its weighted average ,smoothing the image In this scenario, we will be using a Gaussian kernel of 5x5 dimensions:

Image with noise reduced:

Figure 8 Edge Detection

An edge is an area in a picture where the intensity of nearby pixels abruptly shifts. A sudden change indicated by a strong gradient, whereas a weak shift is represented by a weak gradient. So, in a sense, a picture is indeed a stack of matrices with intensities rows and columns. According to this, we may also define an image in 2D coordinates, where the x-axis navigates the width (columns) and the y-axis traverses the height (rows) of an image. The Canny function measures the change in intensities in relation to neighbouring pixels by taking a derivative on the x and y axes. To put it another way, we are calculating the gradient—a change in brightness—in every direction. The sharpest gradients are then traced using white pixels.

Canny = cv2.Canny (blur, 50, 150)

The image is displayed after employing the Canny Function:

Figure 9 Region of Interest

The image's size is chosen in such a way that road lanes are visible and identify them as our region of interest, or the triangle. Subsequently, a mask of the similar dimension as the image is generated, which is essentially an array consisting of zeros. Considering that the masked triangular dimensions are filled with an intensity of 255 so that the dimensions of our region of interest are white.

Below is the image of the mask

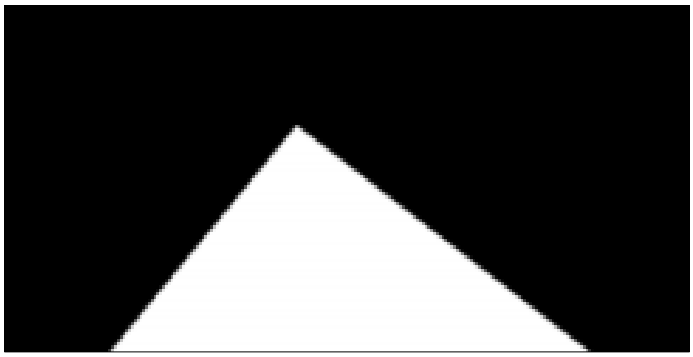

Figure 10 Mask

In order to mask the region of interest, we will now perform a bitwise AND operation on the canny picture.

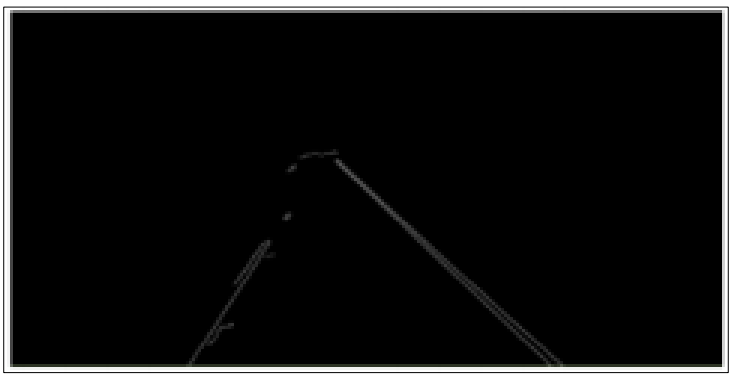

Figure 11 Hough Transform

In order to locate straight lines within image and subsequently locate the lane lines, we use a Hough transform technique. A line is represented by the equation y= mx + b. And the line's slope is just a climb over run. If the y intercept and slope are known, the line is depicted as a single dot in Hough Space. Several lines that can traverse through this point, each with a distinct b and m value. There are several lines that can cross each point independently, each with a distinct slope and y intercept. One line, though, connects the both the points. This can be inferred by looking at the

sufficient space for the intersection, as the Hough Space intersection shows the m and b values of the matching line to cross both points. We first create a grid in our Hough space to help us locate the lines. Each grid bin represents the slope and y intercept of the line. For each Hough Space bin intersection, we place a vote inside bin it belongs to. Our line will be drawn from the bin with the most votes. But we all acknowledge to the fact that a vertical line has an infinite slope. Instead of cartesian coordinates, we will apply polar coordinates to describe vertical lines. As a result, our line's equation becomes.

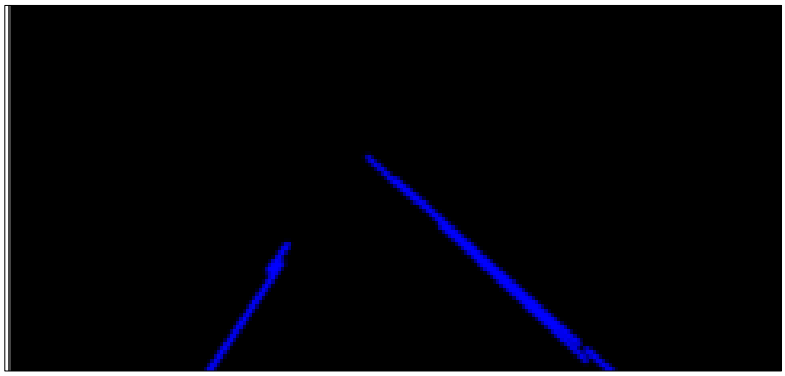

Figure 12 Zero Intensity Image

Figure 13 Combined image

4. Hardware

Raspberry Pi OS: Debian-based Raspberry Pi OS is an operating system created specifically for the Raspberry Pi devices. It resembles operating systems like macOS and Microsoft Windows in appearance. The top menu toolbar has an application menu as well as shortcuts keys to Terminal, Chromium, and File Manager. The right side contains a volume control, Bluetooth menu, a digital clock and a Wi-Fi menu. It is an open-source operating system that can be downloaded for free from the Raspberry Pi website.

4.1. Arduino IDE

An open-source software tool created by Arduino called the Integrated Development Environment (IDE) makes it easy to program the Arduino microcontroller. The Arduino IDE uses unique code structuring principles to accommodate mixed syntax from the programming languages C and C++.

4.2. OpenCV

The Open Source Computer Vision (OpenCV) library is a free and open-source collection of hardware, software, and programming tools for real-time computer vision. For real-time computer vision and machine learning, OpenCV has more than 2500 optimised algorithms. It can recognise items, classify human actions, detect and differentiate faces, identify things, track the movement of objects, and more. It has interfaces for C++, Python, Java, and MATLAB and supports Windows, macOS, Linux, and Android.

5. Results

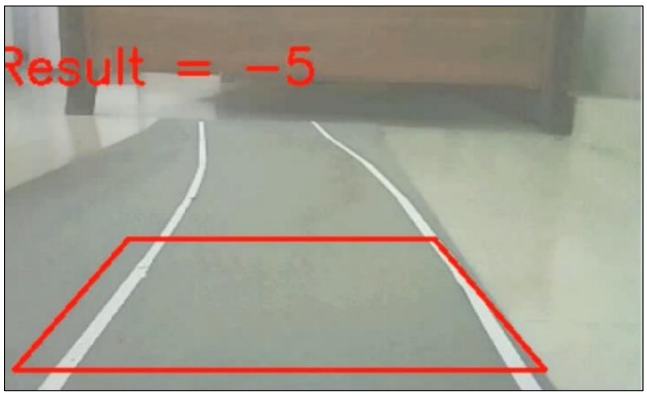

Figure 14 Left side turn with -5

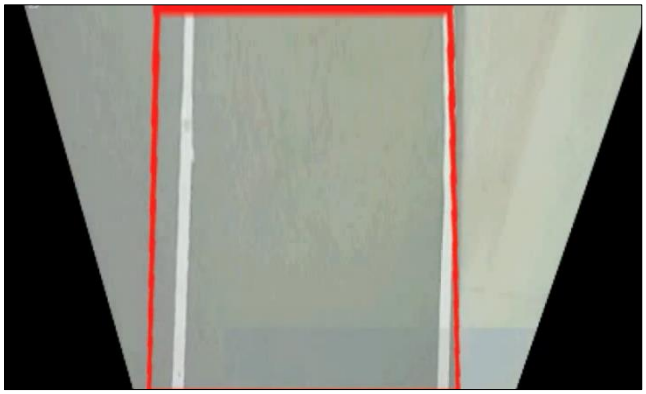

Figure 15 Straight Line direction

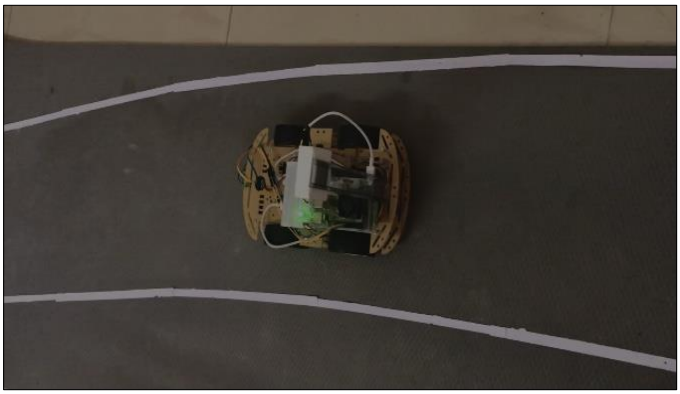

Figure 16 Right side turn

6. Conclusion

In this research method, edge detection was achieved by implementing the OpenCV library and its functions, such as the Canny Function. Next, we created a mask without intensity and used a bitwise operation to map the region of interest. The Hough Transform method was then used to detect straight lines in the image and to detect band lines. We used polar coordinates because rectangular coordinates do not provide us with sufficient slope of the vertical and horizontal lines. Finally, we combined the band image with our zero-potential image to show the band lines.

Compliance with ethical standards

Acknowledgments

We are very much thankful to prof. Sandip Shinde, HOD, Department of Computer Engineering, VIT, Pune for his encouragement and guidance to carries out this research work.

Disclosure of conflict of interest

The authors whose names are listed certify that they have NO affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. All the above information is true and correct

References

[1] https://dev.to/divshekhar/lane-detection-opencvpython-0i6

[2] https://www.intertraffic.com/news/autonomous driving/ut nomous-vehicle-technology-2020/

[3] https://www.irjet.net/archives/V2/i3/Irjet v2i3270.pdf

[4] https://www.hindawi.com/journals/ddns/2012/2731 64/

[5] https://www.researchgate.net/publication/4356226_ Realtime_lane_detection_for_autonomous_vehicle s

[6] https://www.irjet.net/archives/V6/i1/IRJET V6I1245.pdf

[7] 3_Review_of_Lane_Detection_and_Tracking_Alg orithms_in_Advanced_Driver_Assistance_System

[8] http://www.ijcim.th.org/past_editions/2019V27N1/ 27n1Page58.pdf

[9] https://ieeexplore.ieee.org/document/790529810. https://www.kdnuggets.com/2017/07/road-lane line-detection-using-computer-vision-models.html